Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A computer system, comprising:

a pipelined, simultaneous and redundantly threaded ("SRT") processor comprising a fetch unit that further comprises a branch predictor and a counter, said counter causes instructions in a trailing thread to be fetched after corresponding instructions in a leading thread are fetched; and

~~an I/O controller coupled to said processor;~~

~~an I/O device coupled to said I/O controller; and~~

a main system memory coupled to said processor;

wherein said SRT processor processes a set of instructions in ~~a~~ the leading thread and also in ~~a~~ the trailing thread and the SRT processor speculates on the outcome of branch instructions in the leading thread using the branch predictor, but wherein the SRT processor does not speculate on the outcome of branch instructions in the trailing thread and instead uses the actual outcome of branch instructions in the leading thread to predict the outcome of corresponding branch instructions in the trailing thread.

2. (Original) The computer system of claim 1 further comprising a branch outcome queue located in the fetch unit;

wherein the actual outcomes of branch instructions in the leading thread are placed in the branch outcome queue.

3. (Original) The computer system of claim 2 wherein the fetch unit accesses the branch outcome queue and not the branch predictor to predict the outcome of branch instructions in the trailing thread.

4. (Original) The computer system of claim 2 wherein the branch instruction queue is a FIFO buffer.

5. (Original) The computer system of claim 2 wherein the individual branch outcome entries in the branch outcome queue comprise a program type identifier and a target address for the location of the next instruction in the thread to be executed.

6. (Original) The computer system of claim 2 further comprising a register update unit;

wherein the register update unit is configured to hold instructions in a queue until the instructions are executed and retired by the SRT processor and wherein the outcomes of branch instructions in the leading thread are not placed in the branch outcome queue until the branch instructions retire from the register update unit.

7. (Original) The computer system of claim 2 further comprising a slack counter located in the fetch unit;

wherein the slack counter is configured to maintain an approximately constant number of instructions of separation between corresponding instructions in the leading and trailing threads.

8. (Currently Amended) The computer system of claim 3 wherein if the branch outcome queue becomes full, execution of instructions in the ~~first~~ leading thread is temporarily halted to prevent more branch outcomes from entering the branch outcome queue; and

wherein if the branch outcome queue becomes empty, execution of instructions in the ~~second~~ trailing thread is temporarily halted to allow more branch outcomes to enter the branch outcome queue.

9. (Currently Amended) A pipelined, simultaneous and redundantly threaded ("SRT") processor, comprising:

a fetch unit that fetches instructions from a plurality of threads of instructions;

a program counter configured to assign program counter values to instructions in each thread that are fetched by the fetch unit;

an instruction cache coupled to said fetch unit for storing instructions to be decoded and executed; ~~and~~

decode logic coupled to said instruction cache to decode the type of instructions stored in said instruction cache; and

a counter;

wherein said processor is configured to detect transient faults during program execution by executing instructions in at least two redundant copies of a program thread and wherein misspeculation caused by incorrectly predicting the outcomes of branch instructions in a second program thread is avoided by using the actual outcomes of branch instructions in a first program thread to correctly determine the outcome of branch instructions in the second program thread; and

wherein said counter causes instructions in the second program thread to be fetched after corresponding instructions in the first program thread are fetched.

10. (Original) The SRT processor of claim 9 wherein instructions in the first program thread execute in advance of the corresponding instructions in the second program thread thereby creating a slack of instructions between the first and second program threads and wherein said slack is sufficient to allow the SRT processor to resolve any misspeculation in the first program thread prior to providing correct branch outcome results to the second program thread.

11. (Currently Amended) The SRT processor of claim 10 wherein said fetch unit comprises:

a slack counter configured to maintain a target number of instructions of separation between corresponding instructions in the ~~leading~~ first and ~~trailing~~ second threads.

12. (Original) The SRT processor of claim 9 wherein said fetch unit comprises:

a branch predictor for predicting the outcomes of branch instructions in the first program thread; and

a branch outcome queue for storing the actual outcomes of branch instructions in the first program thread;

wherein the actual outcomes of branch instructions in the first program thread are stored in the branch outcome queue after the branch instructions in the first program thread are retired by the SRT processor; and

wherein the fetch unit uses the branch outcome queue and not the branch predictor to predict the outcomes of branch instructions in the second program thread.

13. (Original) The SRT processor of claim 12 wherein the SRT processor is an out-of-order processor capable of executing instructions in the most efficient order, but wherein branch instructions are executed in the same order in both the first and second program threads.

14. (Original) The SRT processor of claim 13 wherein the branch outcome queue is a FIFO buffer and data is transmitted to and from the buffer using an error correction technique.

15. (Original) The SRT processor of claim 12 wherein the individual outcomes stored in the branch outcome queue comprise:

a program type classifying the branch instruction; and

a target address corresponding to the instruction to be executed immediately following the branch instruction;

wherein during execution of the second program thread, the SRT processor may identify the appropriate branch instruction using the program counter value and may also fetch instructions ahead of the branch instruction using the target address.

16. (Original) The SRT processor of claim 12 wherein if the branch outcome queue becomes full, the first thread is stalled to prevent more branch outcomes from entering the branch outcome queue; and

wherein if the branch outcome queue becomes empty, the second thread is stalled to allow more branch outcomes to enter the branch outcome queue.

17. (Currently Amended) A method of predicting branch instructions in ~~an~~ a simultaneous and redundantly threaded ("SRT") processor which can fetch and execute a set of instructions in two separate threads so that each thread includes substantially the same instructions as the other thread, one of said threads being a leading thread and the other of said threads being a trailing thread, the method comprising:

training a branch predictor to store predicted outcomes from branch instructions in the leading thread;

probing the branch predictor to predict outcomes of future executions of branch instructions in the leading thread;

storing actual outcomes of branch instructions in the leading thread in a branch outcome queue;

probing the branch outcome queue to predict outcomes of corresponding branch instructions in the trailing thread; and

decrementing a counter when the leading thread commits an instruction and incrementing the counter when the trailing thread commits an instruction.

18. (Original) The method of claim 17 further comprising:

executing the branch instructions in the leading and trailing threads in program order.

19. (Original) The method of claim 18 further comprising:

storing the actual outcomes of branch instructions in the leading thread in the branch outcome queue after the branch instructions retire;

wherein the outcomes are identified by a branch identifier and a target address signifying the subsequent instruction to be executed as a result of the outcome of the execution of the branch instruction.

20. (Original) The method of claim 18 further comprising:

using a FIFO buffer as the branch outcome queue;

wherein if the buffer becomes full, the leading thread is stalled to prevent more branch outcomes from entering the buffer; and

wherein if the buffer becomes empty, the trailing thread is stalled to allow more branch outcomes to enter the buffer.

21. (Original) The method of claim 18 further comprising:

transmitting data to and from the branch outcome queue using an error correction technique.